COSC460

RESEARCH PROJECT

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF CANTERBURY

SUPERVISOR: PROF JP PENNY

MONITORING OF DISK PERFORMANCE ON THE PRIME COMPUTER SYSTEM

GERARD DUNNE

OCTOBER 1982

Table of Contents

1 INTRODUCTION

A study has been made of the performance of the PRIME computer system at Canterbury University. The PRIME is essentially an interactive system. The performance parameter of most interest to users sitting at a terminal is response time, defined by Abrams [1] as,

> "the elapsed time from the last user keystroke until the first meaningful character is displayed at the user's terminal".

A system's responsiveness is degraded, often quite seriously, by resource bottlenecks. Thus a major aim of the study has been to look for possible bottlenecks which might be having such an effect on response time performance.

An initial general study showed:

(1) the shortcomings of software monitoring,
(2) that disk IO seemed to be a problem.

It was therefore decided to build a disk monitor to concentrate monitoring on this area of system performance. Limitations were encountered using this monitor and from the experience gained in monitoring the disk, the specifications for an ideal disk monitoring tool were developed.

2 BACKGROUND

The evaluation of the PRIME began as a limited study with the intention of producing a report to management. Monitoring was done using the software monitoring tool USAGE. The main area of concern was how the system resources were being utilized. The results of the monitoring seemed to indicate that the system had plenty of spare capacity. However it was later discovered that USAGE was not completely reliable.

2.1 MONITORING

The tool employed, USAGE, is a modified version of the PRIME software monitor. The aim was to get an overall 'workload profile' of the system and then to look for correlations which might give information on system behaviour, and for possible bottlenecks.

One of the first problems encountered was that of getting reliable and useful results.

It was found that USAGE had a number of errors and results were often found to be quite inaccurate. The running of USAGE itself also has some influence on results. However, apart from running it for very short intervals (less than 30 seconds according to the documentation) the effect of this interference is not significant.

Software monitors (and USAGE is no exception) can produce prodigious amounts of data. The performance analyst has the task of reducing this to some meaningful form. The large amount of data produced is only really comprehensible when one or two key measures are extracted and displayed, say in a graph, along with some key statistics such as mean or variance.

Despite these difficulties, by employing USAGE some useful information could be obtained about the PRIME system. The key areas to be looked at were:

(1) CPU utilization
(2) IO utilization
(3) Page Fault rate
(4) Number of Active Users

Specific interest lay in seeing how the system behaved in relation to the Number of Active Users, an 'Active User' being defined by USAGE as any user who has used either CPU or IO time since the last sampling by the monitor. This definition is not strictly correct. The figures gained depend very much on the length of the interval. The longer the interval the greater the Number of Active Users will appear to be, since any user who has used CPU or IO resources over the period will

be counted. Thus USAGE gives an over-estimate of the Number of Active Users.

2.2 RESULTS

2.2.1 CPU UTILIZATION

A first observation was that there was no significant correlation between CPU utilization and the Number of Active Users. The explanation for this is found when one considers the way in which the scheduler queues are organised (see Appendix B). Essentially the high utilizations were being caused by large CPU-intensive jobs (such as MATH, CHEM, PHYS) soaking up spare CPU time not being used by short tasks. That is, the value of CPU utilization was heavily influenced by the presence (or absence) of one or two large jobs. When the effect of these was eliminated, it seemed that the CPU's performance was not having a major effect on the overall system performance.

It was also noted [6] that, discounting these large tasks:

(1) a broad correlation existed between CPU utilization and the Number of Active Users,
(2) CPU utilization seldom rose above 60%.

2.2.2 IO UTILIZATION

Observations regarding IO were however quite different. There certainly was a correlation between Number of Active Users and IO utilization as one would expect, but there was a very much stronger correlation between IO utilization and the Page Fault rate. Whenever Page Faults rose, IO also rose.

Thus the following conclusions were reached [6]:

(1) Page Faults were constituting a large proportion of the total IO.
(2) IO utilization figures according to USAGE never rose above about 30% so it was assumed to be even less of a problem than the CPU.
(3) Page Faults seemed to be within acceptable limits [15,16].

2.3 RELIABILITY

These results highlight a major difficulty in monitoring, that of ensuring that measurements are reliable. It was later discovered by Freeth [7] that USAGE was not including the seek times for the disks correctly. Thus what at first had appeared as an under-utilized resource was subsequently found to be quite heavily utilized. When the metering was corrected and further monitoring was done, the IO utilization was as high as 90% (out of a theoretical maximum approaching 200%, but 3 times the former high experienced). At this stage we suspected that disk IO might indeed be the system bottleneck, and that further monitoring was needed.

3 THE DISK MONITOR

Having decided to concentrate on monitoring the disk system, a tool was needed that would collect more information than USAGE did on the utilization of the disks. It was also desirable to include some data reduction so that data would be given in an easily understood form (for example, as graphs of the day's activity). This, coupled with the gathering of a comprehensive set of performance indices, was included in the design of the disk monitoring tool.

3.1 DESIGN

The first step in the design of the tool was to decide what parameters of disk performance to meter. There were two difficulties. First, while some of the meters, which gather the data about the system, had been corrected after our discoveries with USAGE, others were still not accurate. Second, only a limited number of items were measurable with the meters available.

The tool had to be designed within these limitations. Having found out what was available, it was then necessary to decide which of these indices would be useful in determining just how the disks were performing. The utilization could be broken down along the following lines:

(1) user and system IO (including paging),
(2) disk1 and disk2 IO (including paging),
(3) IO (excluding paging) and paging.

To distinguish between input-output excluding paging and input-output including paging here, the acronym IO is used for the former, and IP for the latter.

Since no particular data about the disk workload was being sought at this stage, it was decided to monitor everything possible, and then to isolate particular areas of interest for further study and analysis.

For some of the parameters measured, it is possible to get the time taken in ticks (there are 330 ticks/second), while for others the number of accesses made is given. So with these units available, the following were measured:

In terms of time:

(1) disk1,disk2 IP
(2) user,system IP
(3) total IP

In terms of accesses:

(1) disk1 IO
(2) disk2 IO,PF,IP
(3) total IO,IP

For examples of the monitor's output see Appendix C.

3.2 MONITORING

The monitor was run simultaneously with USAGE over a number of days. One of the most difficult tasks is to validate the results, a problem of course for any such monitoring. The only real way to achieve validation is to use the monitor continuously over a long period of time. Through continual checking, confidence is built up in the validity of the metering [13].

USAGE also aided in the validation process since the results it produced could be checked against those of the disk monitor. If they were in agreement then at least the disk monitor algorithm was correct. This did not of course validate the meters since USAGE was using the same ones. (In fact some of the software from USAGE was incorporated, modified, into the disk monitor).

Another reason for running USAGE was to have available the values of the other meters over the metering period in case they proved useful to explain results.

3.3 RESULTS

The disk monitor produces results in both tabular and graphical form.

The most significant result is the imbalance found in the utilization of the two disks. The metering done (see Appendix C), showed that the number of accesses for disk2 is 7 times that for disk1. Clearly this is an unsatisfactory situation.

The two disks work in parallel with a channel each. If one disk is doing considerably more work than the other, then little advantage is being taken of this parallelism to overlap the disk usage and thus reduce the total time for which either disk is busy.

Having discovered this imbalance, information was sought on the layout of the disks to explain what causes it. Information provided by [4] indicated that the files are distributed as follows:

(1) disk1: ~30% of the user files
(2) disk2: system files
paging area
~70% of the user files

A more detailed breakdown is given in Appendix D.

It is apparent that the distribution of files across the disks is hopelessly unbalanced. The reason is that when the PRIME was first installed, disk1 was not functioning correctly and so, to run the system, all the files were put on disk2.

Computer Centre staff intend to rectify this over the summer vacation.

The other result of note is that paging constitutes only about 25% of IO accesses done, and so is not really the problem it was thought to be earlier in the year.

3.4 RECOMMENDATIONS FOR IMPROVEMENT

The disks are at present unbalanced in terms of their file distribution. This in turn gives rise to an imbalance in the workload of the disks. We want to rectify this situation to minimise the total time for which either disk is busy. One way to do this is to balance the file positioning across the two disks [2,16]. This should increase the overlap of accesses to the disks and thus reduce the total time for which either disk is busy.

There are a number of strategies for repositioning.

One solution would be to interleave the records of each file on each disk, in much the same way that memory is interleaved on odd and even memory address boards, to achieve maximum overlap.

A more practical solution would be to determine the frequency and order of access to files and then distribute them so that each disk would do an equal amount of work with as much as possible being done in parallel (hence the monitoring of the order in which accesses are made). This however requires monitoring of a much more detailed nature than has been done on the PRIME.

The next best option would be to split each of system, paging, and user areas evenly, putting half of each on each disk. However the disk scanning software on the PRIME relies on

physical addresses to access system and paging areas. Thus it would be rather awkward to shift them.

For the PRIME, the simplest solution is to shift a suitable number of user files onto disk1 in order to strike the balance required.

The following calculations, based on the results of this study lead to the recommendation that:

75% of user files be placed on disk1
25% of user files be placed on disk2

From measurements made, the ratio of USER-IP to SYSTEM-IP is roughly,

$$\frac{\text{USER-IP}}{\text{SYSTEM-IP}} = \frac{9}{1}$$

and the overall ratio of IO to PF is,

$$\frac{\text{IO}}{\text{PF}} = \frac{3}{1}$$

Some system pages are locked into memory, but system tasks are run more frequently than user tasks, so let us assume that user and system tasks do roughly equal proportions of IO and PF. Then the disk transfers would be divided, as follows:

USER-IO	67.5%
USER-PF	22.5
SYSTEM-IO	7.5
SYSTEM-PF	2.5

We want 50% of the file accesses to be made to each disk. System and paging areas remain on disk2 so the following file accesses will be all to disk2,

USER-PF	22.5%
SYSTEM-IO	7.5
SYSTEM-PF	2.5
TOTAL	32.5%

This leaves the USER-IO accesses to be distributed across the two disks. To balance the disks we need an additional 17.5% of the file accesses to be to disk2,

$$\frac{17.5}{67.5} = \sim 25\% \text{ of USER-IO}$$

This leaves 50% of the file accesses on disk1,

$$\frac{50}{67.5} = \sim 75\% \text{ of USER-IO}$$

Thus we want 25% of USER-IO accesses to be to disk2 and 75% to disk1. Not all user files have an equal probability of being accessed so we must ensure that the positioning of the files is done carefully with this in mind [16].

While it is not possible to determine the amount of improvement that will result from the redistribution of the files we can calculate the maximum possible. If we make the assumption that the system is presently achieving the maximum overlap possible with its current imbalance and that it will do the same to achieve complete overlap when balanced then the following improvements will be obtained:

(1) a reduction in the time that either disk is busy
Currently the ratio of disk2 to disk1 activity is,

$$\frac{\text{disk2}}{\text{disk1}} = \frac{7}{1}$$

So 8 requests take 7 time units.
When the disks are balanced the ratio will be,

$$\frac{\text{disk2}}{\text{disk1}} = \frac{1}{1}$$

So 8 requests will take 4 time units.
Thus the improvement in time taken to service requests would be,

$$\frac{7 - 4}{7} = \frac{3}{7} = \sim 43\%$$

- a 43% reduction in the total time that either disk is busy.

(2) an increase in the disks' capacity to service requests
Currently the ratio of disk1 to disk2 activity is,

$$\frac{\text{disk2}}{\text{disk1}} = \frac{7}{1}$$

So 8 requests take 7 time units.

When the disks are balanced the ratio will be,

$$\frac{\text{disk2}}{\text{disk1}} = \frac{1}{1}$$

So 14 requests will take 7 time units.
Thus the improvement in the amount of IO that can be serviced is,

$$\frac{14 - 8}{8} = \frac{6}{8} = 75\%$$

- a 75% increase in the amount of IO that would be serviced for the same level of system performance.

These are the maximum gains possible given the assumptions above, and the actual gains are likely to be significantly smaller, though by how much it is not easy to estimate.

Follow up work in a number of areas is also recommended.

First, the disk monitoring should continue, to see what improvement has actually been achieved and to make further adjustments (see 4.1) as necessary. This will also provide some validation or otherwise for the disk monitoring tool. It may be possible to improve the tool and add to it if more metering is made available in further releases of the system software (for example, metering of Page Faults for each user as is promised for Revision 19).

In addition, further tools could be developed to look into other areas of disk performance [16], as described in the next section.

4 THE IDEAL DISK MONITOR

The aim of disk monitoring is to find ways of tuning the disk system so that it works as efficiently as possible. Tuning has a twofold benefit. The current workload is done more quickly thereby improving the level of responsiveness of the system, and the total throughput capacity of the disk subsystem is increased.

4.1 BACKGROUND TO DESIGN

To make the disk work more efficiently let us first consider what components make up disk activity. If we have n transfers which require t_i seconds each, then the total time to process the workload consecutively, that is without overlap, is

$$T_c = \sum_{i=1}^{n} t_i$$

Now t_i can be broken down into seek time s_i, latency l_i, and transfer time tt_i, so,

$$\sum t_i = \sum (s_i + l_i + tt_i)$$

Transfer time for a particular disk is fixed and so cannot be improved. Latency, assuming all disk accesses are independent, will be a uniform random function varying between zero and the rotational time of the disk with an average of half the rotational time [2]. While disk accesses are not independent, there is relatively little that can be done to reduce latency.

Seek time is the major component in disk accessing. On the PRIME for example average access times are, s_i=30ms, l_i=8.3ms, tt_i=1.7ms. Seek time can be reduced in a number of ways. Three methods are discussed in Atwood et al [2]:

(1) seek overlap
(2) seek arm scheduling
(3) file repositioning

Their results showed that seek arm scheduling, that is, queueing requests for disk accesses in the order in which the records to be accessed are positioned across the disk, is unlikely to produce much improvement. However file repositioning on a single disk can improve performance and even greater improvements can be gained from overlapping seeks.

Reed [16] discusses in some detail, strategies for positioning files on disks to minimise arm movement.

Overlap can be considered for a general system with p disks. If T_ρ is the total time for which any of the disks are busy then the improvement factor I, due to overlap is

$$I = T_c / T_\rho$$

as shown by Hellerman [9]. In order to maximise the overlap we want each disk to be doing an equal share of the workload and we want this to be done with as much concurrency as possible. Thus file distribution should be done with the aim of achieving these objectives.

A further technique for fine tuning the disks is to change the relative sizes of the associative buffers [10]. For instance in a two disk system, if one disk is found to be doing a little more work than the other then we can increase the size of its associative buffer area at the expense of the other disk's area. This however is only a fine tuning measure. The disks need first to have a balanced file distribution.

To summarise, the efficiency of a disk subsystem can be improved by the following steps:

(1) distribute files across disks to maximise the overlap of disk activity,

(2) position files on each disk so as to minimise seek time,

(3) adjust associative buffers to compensate for small differences in disk workloads.

4.2 SPECIFICATIONS

The approach to the design of the ideal disk monitor has been a fairly general one. The items it includes are relevant to most disk systems. Unfortunately on the PRIME, many of the items that would be desirable are not metered. In addition there are indices which would be useful to obtain but are in practice difficult to monitor, such as the extent and nature of the overlap, and the efficiency of the file placement.

Software monitors interfere with what they monitor. The more detailed the monitoring the greater the interference. Therefore a point is reached where it is desirable to use a hardware monitor, especially for the collection of data at the microscopic level.

The ideal monitor could undoubtedly produce a vast amount of data. For any particular system it would be necessary to do some judicious data selection at the time of measurement, depending of course on what is being looked for. Short of having something definite in mind, it is best to start by monitoring as much as possible. From this base, areas of

interest can be isolated for further detailed study.

Data reduction is also important in monitoring. Graphs and scatter diagrams are particularly useful for determining system trends and correlations.

The ideal monitor would of course not interfere at all with the system being monitored. However this is simply not possible for a software monitor.

The chief parameters we want to be able to meter are:

(1) seek time
(2) latency
(3) transfer time
(4) total time
(5) number of accesses

On the PRIME this would be done for the following activities:

(a) diskl IO
(b) disk2 IO,PF
(c) user IO,PF
(d) system IO,PF

Other items which might be useful are:

(6) the frequency with which files are accessed
(7) the order in which files are accessed
(8) the pages which are swapped in and out of memory
(9) the behaviour of the IO queues

5. CONCLUSIONS

This study has concentrated on disk activity as a contributing factor in overall system performance. The principle reason why system performance is hindered on the PRIME at the moment, appears to be that the disks are not being utilized efficiently. In particular, little advantage is being made of the parallelism possible in the operation of the two drives.

It has not been possible to measure directly the effect that such inefficiency has had on system performance. However it is clear that the present situation could be improved by tuning the disks as described.

Limitations encountered in the monitoring have lead to the specification of an ideal disk monitor which would provide a far better understanding of the disk system. It has also been recommended that:

(1) A continuous programme of system performance monitoring be run.
(2) Further tools be developed as needed, to evaluate other factors contributing to system performance.

Such monitoring of system activity is seen as a key step in improving system efficiency and thus leading ultimately to a better service for users.

ACKNOWLEDGEMENTS

I wish to thank my supervisor JP Penny, for ideas and advice in connection with the project, and GD Freeth and CM Brown for assistance on matters concerning PRIME software and hardware.

REFERENCES

[1] ABRAMS MD, 'Techniques for Evaluating the Effectiveness of Interactive Computing Service', PROCEEDINGS OF THE ACM NATIONAL CONFERENCE, 1977, p452

[2] ATWOOD JW et al, 'An Empirical Study of a CDC 844-41 Disk Subsystem', PERFORMANCE EVALUATION 2, 1982, p29

[3] BASTIAN AL et al, 'Characteristics of DASD Use', PROCEEDINGS OF THE CMG XII INTERNATIONAL CONFERENCE', December 1981, p107

[4] BROWN CM, Personal Communication, 1982

[5] BUZEN JP, 'Fundamental Laws of Computer System Performance', PROCEEDINGS OF THE INTERNATIONAL SYMPOSIUM ON COMPUTER PERFORMANCE MODELLING, MEASUREMENT AND EVALUATION, 1976, p200

[6] DUNNE GM, 'Performance Evaluation Report on the University of Canterbury PRIME Computer System', Unpublished Report, April 1982

[7] FREETH GD, Personal Communication, 1982

[8] HARING G et al, 'The Use of a Synthetic Jobstream in Performance Evaluation', THE COMPUTER JOURNAL, August 1980, p209

[9] HELLERMAN H & SMITH HJ Jr, 'Throughput Analysis of Some Idealized Input, Output, and Compute Overlap Configurations', COMPUTING SURVEYS, June 1970, p111

[10] MALLERY RD, 'The Effect of Increasing FIP Buffering on a Heavily Loaded 11/70', Informal Memorandum, Nationwide Data Dialog Inc, Southampton, Pa.

[11] MASSEY UNIVERSITY COMPUTER CENTRE, 'Heavy Usage on the PRIME', LOGON, March 1982, p1

[12] PENNY JP, 'A Simple Model for Input, Output and Compute Overlap', THE AUSTRALIAN COMPUTER JOURNAL, July 1977, p72

[13] PENNY JP & SHEEDY CR, 'Measurement of Response Time Performance in Small Time-Sharing Systems', THE AUSTRALIAN COMPUTER JOURNAL, February 1980, p15

[14] PRIME Administrators Guide, PRIME Computer Inc, Framingham, MASSACHUSETTS, 1980

[15] PRIME Internals Manual PRIME Computer Inc, Framingham, MASSACHUSETTS, 1980

[16] REED ML, 'Managing DASD Performance to Satisfy Workload Requirements', PROCEEDINGS OF THE CMG XII INTERNATIONAL CONFERENCE, December 1981, p119

[17] SHEEDY CR, 'A Synthetic Measure of Response Time', Honours Project Report, Department of Computer Science, University of Canterbury, November 1978

[18] SMITH BJ, 'IO Subsystem Workloads, Measurements and Modelling', PROCEEDINGS OF THE CMG XII INTERNATIONAL CONFERENCE, December 1981, p110

[19] WILHELM NC, 'A General Model for the Performance of Disk Systems', JOURNAL OF THE ACM, January 1977, p14

APPENDIX A USAGE

Sample Output

```
07-OCT-82 10:09:22.30  DIFTIM  599.96
CPTOT   305.38  IOTOT   357.71  RETOT  1374.30  CfgUSR     72
DCPTOT  125.13  %CPU     20.86  DPFCN     1023  PF/sec   1.71
DIOTOT  134.52  %IO      22.42  DIOCN     5566  IO/sec   9.28
DLOCNT   21067  LO/sec   35.11  DLOFCT   13010  DLOSCT   3685
DLOUCT      21  DLOCCT    4351  LM/sec    7.25  %MISS   20.65

%Error    0.50  %Back1   74.65  %Back2    0.00  %Clock    1.30  %Farnt    0.00
%SLMC     0.02  %AMLC     2.42  %MPC1     0.02  %MPC2     0.00  %Ver      0.00
%PNC      0.00  %Spare    0.00  %Disk1    0.02  %Disk2    0.22

 No  LOGNAM      Pf     Mem     CPTIME  DIF-CP    %CPU     IOTIME  DIF-IO     %IO

  1  SYSTEM       0     617      29.12    2.42    0.40      82.38    7.83    1.30
  5                 0       3       4.21    4.21    0.70       0.00    0.00    0.00
  9  COSC00       0     247      55.45   26.89    4.48      66.51   36.92    6.15
 10                 0       1      24.81   24.81    4.14       0.00    0.00    0.00
 14  CCTR07       0     200      26.76   26.17    4.36      36.94   34.63    5.77
 28                 0       3       1.23    1.23    0.21       0.00    0.00    0.00
 29  CHEM13       0      58       5.83    5.83    0.97       9.13    9.13    1.52
 31  EXTL16       0      14       2.75    1.14    0.19       2.33    0.06    0.01
 32  CHEM18       0      42      29.11   21.25    3.54      10.59    3.16    0.53
 57  SYSTEM       0      10       1.31    0.29    0.05       6.05    1.66    0.28
 58  SYSTEM       0      11       1.25    0.28    0.05       3.48    0.80    0.13
 59  SYSTEM       0      15       2.31    0.92    0.15       4.66    1.83    0.31
 60  SYSTEM       0      10       1.22    0.33    0.05       4.92    1.87    0.31
 61  SYSTEM       0      29       1.14    0.26    0.04       3.89    1.10    0.18
 62  SYSTEM       0      34       2.71    1.88    0.31       4.15    1.43    0.24
 63  SYSTEM       0      15       4.84    1.81    0.30      13.58    4.41    0.74
 64  SYSTEM       0       3       1.05    0.22    0.04       2.54    0.00    0.00
 65  COSC30       0      13      12.12    7.38    1.23       5.86    1.62    0.27
 66  COSC30       0      13       4.82    4.19    0.70       4.10    1.67    0.28
 67  COSC46       0      15       0.71    0.04    0.01       3.48    0.00    0.00
 68  COSC46       0      63       1.28    0.94    0.16       5.59    3.84    0.64
 69  COSC30       0      34       1.93    1.93    0.32       2.62    2.62    0.44
 --  <idle>       0       1       0.00    0.00    0.00       0.00    0.00    0.00

 23  Users (22 active,  1 idle,  0 remote)  1451 Pages in use
```

APPENDIX B SCHEDULER QUEUES

Scheduling is done using 7 queues [15]:

WAIT LIST

1. ELIGIBILITY
2. HIGH PRIORITY
3. USER 1 (LOW PRIORITY)
4. LEVEL 3 (LOW PRIORITY)
5. LEVEL 2 (LOW PRIORITY)
6. LEVEL 1 (LOW PRIORITY)
7. LEVEL 0 (LOW PRIORITY)

When a job enters the system for execution it is put in the High Priority queue (HQ). The timeslice for this queue is 3/10 second. If it requires more CPU time after it has run for its timeslice it is placed in the Eligibility queue (EQ). Here it gets up to 2 more timeslices of 3/10 second. If it still requires the CPU it is placed in Low Priority queue 6 (LQ). The timeslice in this queue is 2 seconds.

When it is ready to run a task the scheduler looks first in the HQ then the EQ and then the LQ. The HQ must be empty before jobs in the EQ can run and the EQ must be empty before jobs in the LQ can run.

APPENDIX C THE DISK MONITOR

Calculations of disk parameters were based formally on the results of 3 days monitoring done during the week of Sept 27 - Oct 1, 1982. Each day was monitored over an 8 hour period, from 9am to 5pm, with sampling done at 10 minute intervals. The days monitored were,

Monday, September 27
Wednesday, September 29
Thursday, September 30

These were fairly typical days of activity for this time of year, as confirmed by informal monitoring of a number of other days around this time.

The disk monitor does interfere with the system it is monitoring. It executes about 36 write statements every time it samples. However this is not significant when compared to the 10 000 accesses typically made in a 10 minute interval.

A problem, also experienced with USAGE, occurs when users log in and out. For the disk monitor problems only occur when a system task logs out which happens fairly rarely. If the results are likely to be affected the monitor prints a warning message and the results are ignored in calculating utilization statistics.

Sample Tabular Output

DISK USAGE 07-10-82 10:09:34

ELAPSED TIME = 600.02 SECS

	DISK1-IO	DISK2-IP	USER-IP	SYSTEM-IP	TOTAL-IP
SECONDS	11.90	115.90	107.53	20.27	127.79
PERCENTAGE	2.0	19.3	17.9	3.4	21.3

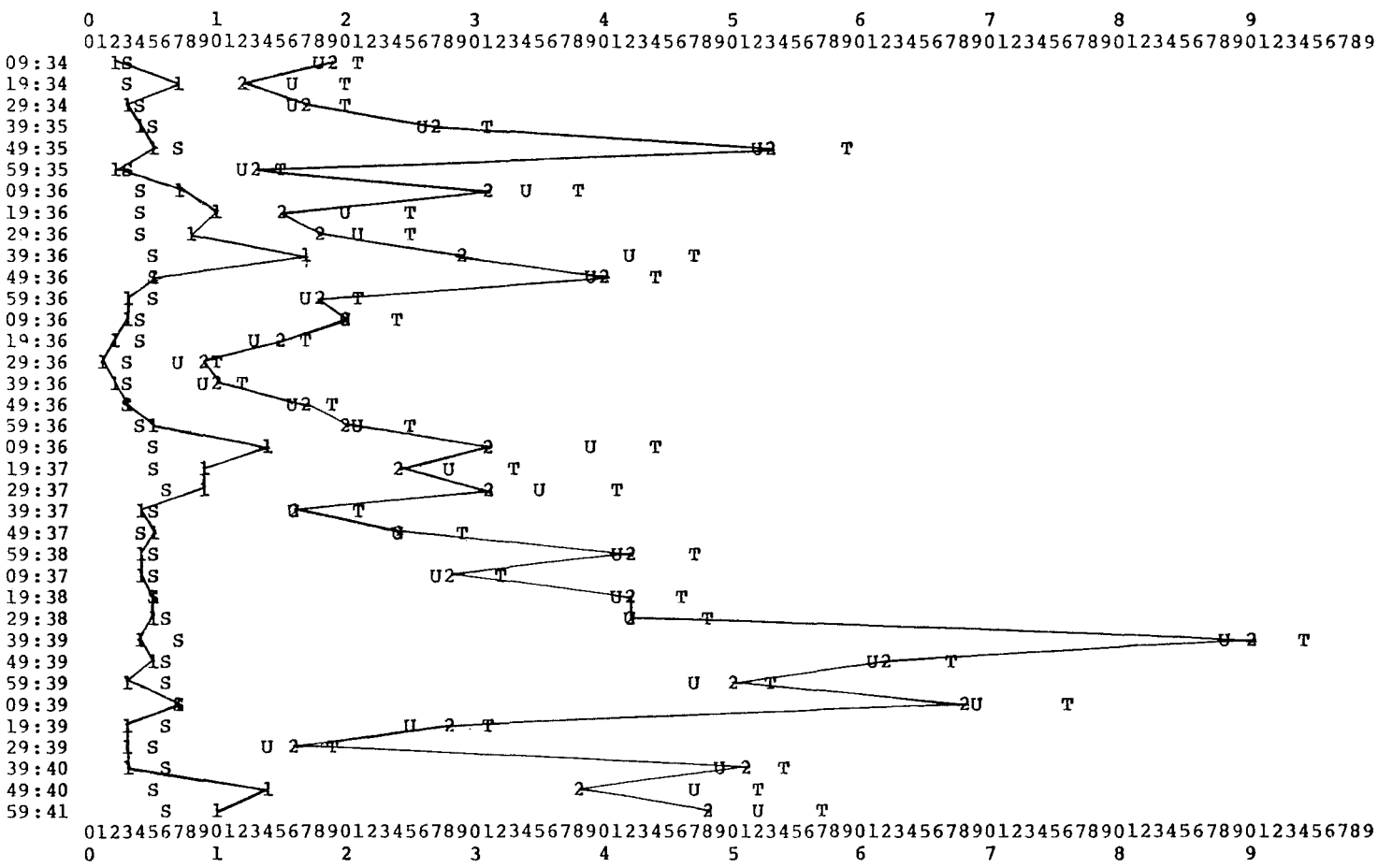
	DISK1-IO	DISK2-IO	DISK2-PF	DISK2-IP	TOTAL-IP
ACCESSES	383.	3985.	968.	4953.	5336.
PERCENTAGE	7.2	74.7	18.1	92.8	100.0

Sample Graphical Output 1

10-82

TIME VS TIME_OF_DAY

KEN DOWN AS FOLLOWS:
 DISK1 IO
 DISK2 IP
 USER   IP
 SYSTEM IP
 TOTAL IP



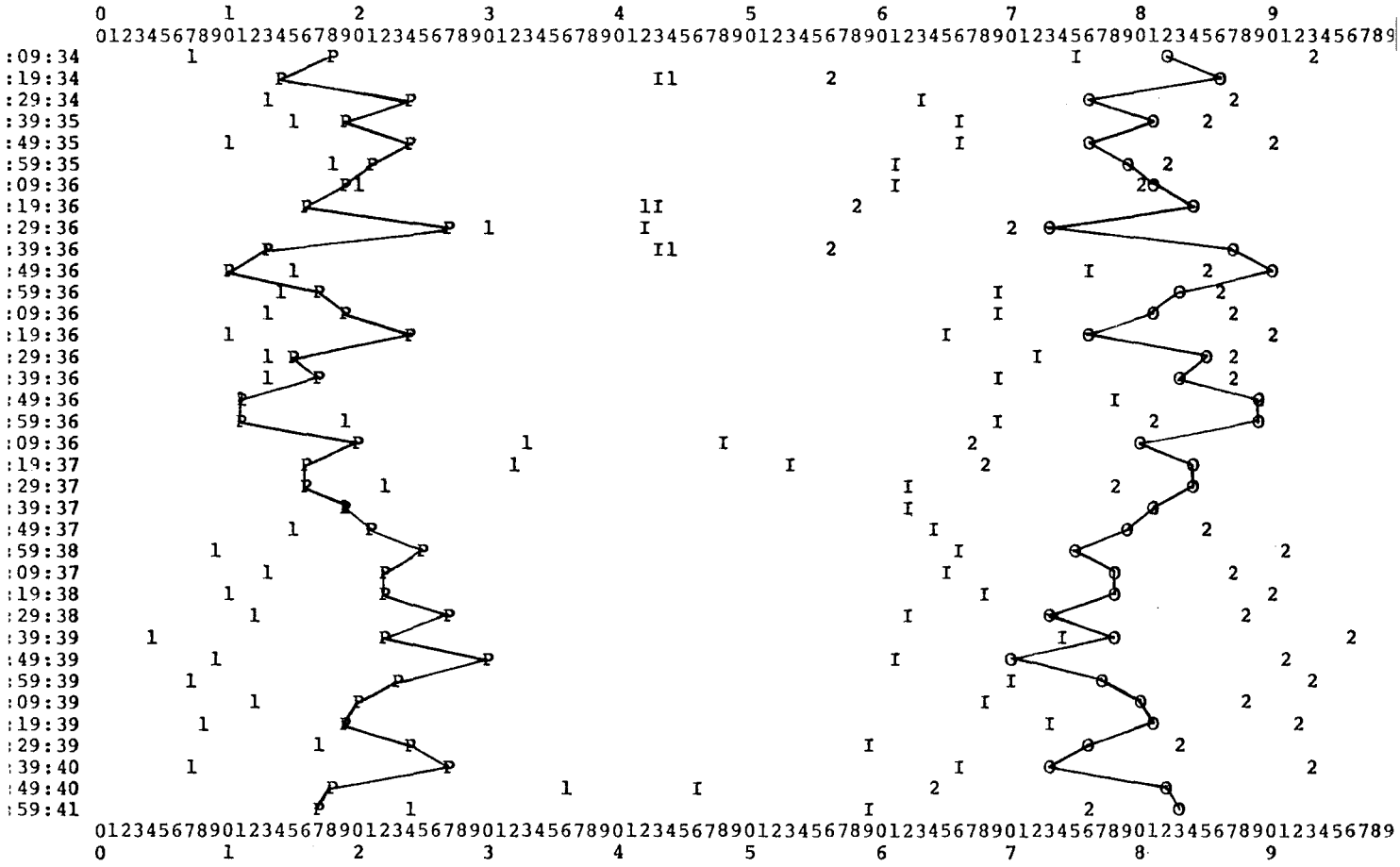
N(DISK1-IO) = 5.6
N(DISK2-IP) = 31.0
N(USER-IP) = 31.8
N(SYSTEM-IP) = 4.7
N(TOTAL-IP) = 36.6

Sample Graphical Output 2

-10-82

CCESSES VS TIME_OF_DAY

)KEN DOWN AS FOLLOWS:
- DISK1 IO
- DISK2 IO
- DISK2 PF
- DISK2 IP
- TOTAL IO



AN(DISK1-IO) = 17.8
AN(DISK2-IO) = 62.6
AN(DISK2-PF) = 19.7
AN(DISK2-IP) = 82.2
AN(TOTAL-IO) = 80.3

APPENDIX D DISK LAYOUTS

The PRIME disk system consists of two drives. In the main body of the report I have referred to two disks,

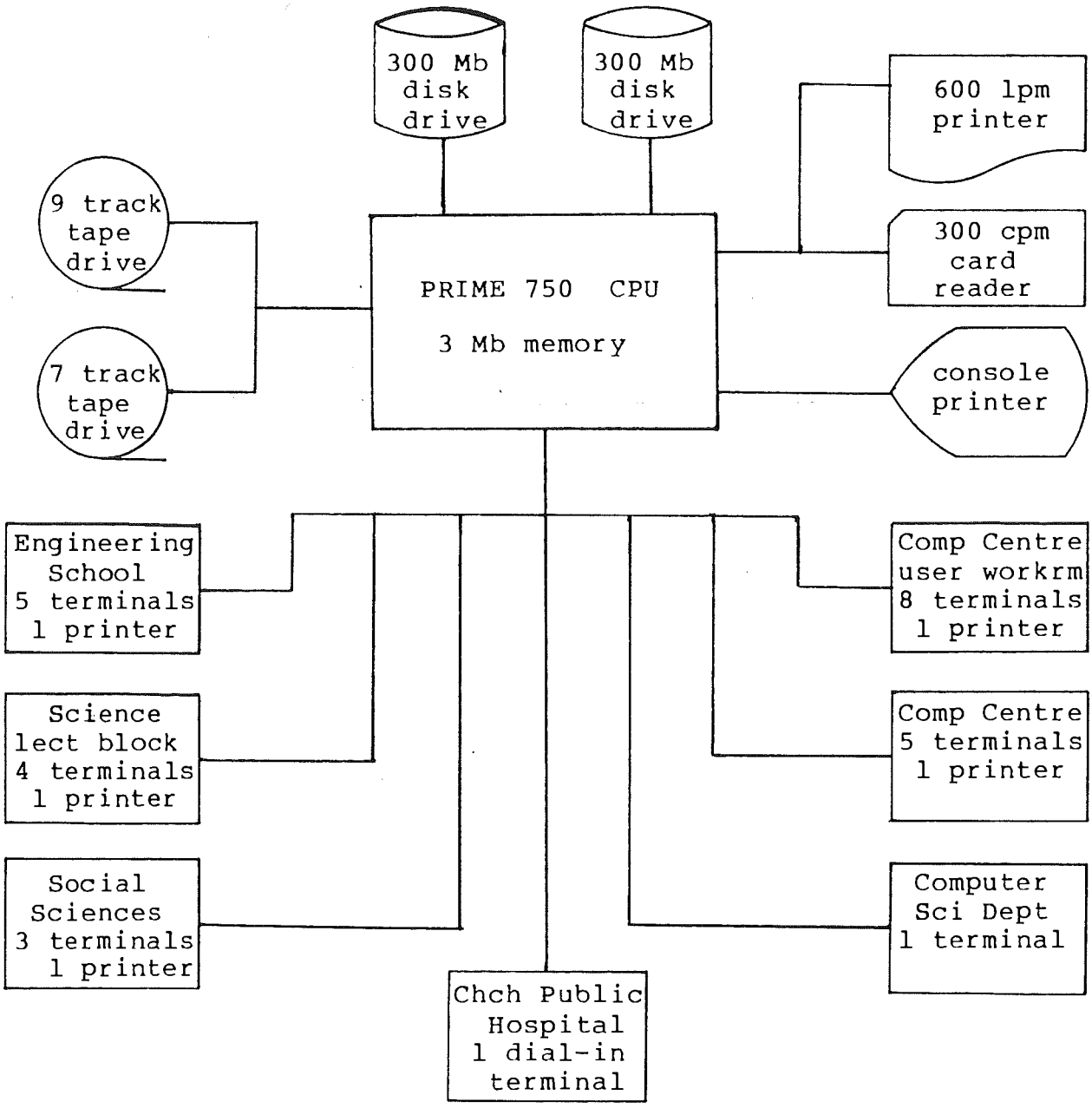
disk1 = drive1
disk2 = drive0

Both of the drives are split up into 3 virtual disks as follows:

	NAME	SURFACES	RECORDS	CONTENTS
drive0	cant00	2	34476	system files
	paging	2	34476	paging area
	cant01	15	258571	~70% of user files
drive1	cant02	2	34476	backup of system files
	cant03	2	34476	backup paging area
	cant04	15	258571	~30% of user files

More details are available on the physical structure of the disks in Appendix D of the PRIME Administrators Guide [14].

APPENDIX E PRIME SYSTEM CONFIGURATION



```
C
C      File system meters
C
       INTEGER*2 G$PFMS,G$MAX2
       PARAMETER G$PFMS=2,G$MAX2=12
       INTEGER*2 G$BUF2(G$MAX2)
       INTEGER*2 G$VER2,G$MBZ2
       INTEGER*4 NPFCN,LOCCT,LOFCT,LOSCT,LOUCT
       EQUIVALENCE
      + (G$BUF2,G$BUF(14)),                 /* QD
      + (G$VER2,G$BUF2(01)),                /* Version
      + (G$MBZ2,G$BUF2(02)),                /* Unused
      + (NPFCN, G$BUF2(03)),                /* Page faults since boot
      + (LOCCT, G$BUF2(05)),                /* Locate reads OR misses ?
      + (LOFCT, G$BUF2(07)),                /* Locate founds
      + (LOSCT, G$BUF2(09)),                /* Locate sames
      + (LOUCT, G$BUF2(11))                 /* Locate useds
C
C      User meters
C
       INTEGER*2 G$USER,G$MAX4
       PARAMETER G$USER=4,G$MAX4=26
       INTEGER*2 G$BUF4(G$MAX4)
       INTEGER*2 G$VER4,US$CLK,US$NAM(16),US$MEM,US$JNK
       INTEGER*4 US$CPU,US$IOT,US$PFC
       EQUIVALENCE
      + (G$VER4,G$BUF4(01)),                /* Version
      + (US$CLK,G$BUF4(02)),                /* Login flags and login clock
      + (US$NAM,G$BUF4(03)),                /* User login name
      + (US$CPU,G$BUF4(19)),                /* CPU time charged to user
      + (US$IOT,G$BUF4(21)),                /* I/O time charged to user
      + (US$PFC,G$BUF4(23)),                /* Page faults charged to user
      + (US$MEM,G$BUF4(25)),                /* Undefined
      + (US$JNK,G$BUF4(26))                 /* Undefined
C
C      Disk meters
C
       INTEGER*2 G$DISK,G$MAX6
       PARAMETER G$DISK=6,G$MAX6=72
       INTEGER*2 G$BUF6(G$MAX6)
       INTEGER*2 G$VER6,G$MBZ6
       INTEGER*4 QWAITS,DMAOVR,DHANGS,DEVIOT(4,4),IOCONT(4,4)
       EQUIVALENCE
      + (G$BUF6,G$BUF(189)),                /* QD
      + (G$VER6,G$BUF6(01)),                /* Version
      + (G$MBZ6,G$BUF6(02)),                /* Unused
      + (QWAITS,G$BUF6(03)),                /* Waits for free queue entry
      + (DMAOVR,G$BUF6(05)),                /* DMA overruns
      + (DHANGS,G$BUF6(07)),                /* Disk controller hangups
      + (DEVIOT,G$BUF6(09)),                /* Per device I/O times
      + (IOCONT,G$BUF6(41))                 /* Per device I/O count
C
C=======================================
C
       LOGICAL*2 TELL,LOGFLT
       INTEGER*2 G$ALL,BUFSZE
       PARAMETER G$ALL=99,BUFSZE=3588
       INTEGER*2 TIMES,HOURS,MINUTS,SECNDS,I,J,K,USER,CLOCK(6),CFGUSR,
      +          CODE,INXUSR,G$BUF(BUFSZE),DISK1,DISK2I,DISK2P,DISK2,
      +          DISKID,USR,SYS,TOTAL
       PARAMETER CFGUSR=72
       INTEGER*4 PERIOD,DELTIM,PGFLTS,SYSID,OSYSID,OPGFLT,BITMAP(CFGUSR),
      +          CURTIM,OLDTIM,ODEVIO(4,4),OCNTIO(4,4),OIOTIM(CFGUSR),
      +          ONPFCN
       REAL*4 TIMMTR(2,5),ACCMTR(2,5),DIFTIM,BLANK,ONE,TWO,TOT,U,S,II,
      +       P,O,TIMAV(5),ACCAV(5)
       REAL*8 XLOGNM
       EQUIVALENCE (XLOGNM,US$NAM)
C
CCCCC  INITIALISATION
C
       CALL TNOU('IDISK MONITOR Version 1.03',26)
       BLANK=' '
       ONE='1'
       TWO='2'
       TOT='T'
       U='U'
       S='S'
       II='I'
       P='P'
       O='O'
       PERIOD=0
       TIMES=0
       TELL=.FALSE.
       LOGFLT=.FALSE.
       OLDTIM=0
       SYSID=0
       DO 40 I=1,CFGUSR
       BITMAP(I)=0
40     OIOTIM(I)=0
       DO 30 I=1,5
       TIMAV(I)=0
30     ACCAV(I)=0
```

```
C
      CALL TIMDAT(CLOCK,6)
C
      WRITE(7,70)CLOCK(2),CLOCK(1),CLOCK(3)
70    FORMAT(1X,A2,'-',A2,'-',A2//
     +       1X,'%IOTIME VS TIME_OF_DAY'/
     +       1X,'-----------------------'//
     +       1X,'BROKEN DOWN AS FOLLOWS:'/
     +       1X,'1 - DISK1  IO'/
     +       1X,'2 - DISK2  IP'/
     +       1X,'U - USER   IP'/
     +       1X,'S - SYSTEM IP'/
     +       1X,'T - TOTAL  IP'//)
      WRITE(8,80)CLOCK(2),CLOCK(1),CLOCK(3)
80    FORMAT(1X,A2,'-',A2,'-',A2//
     +       1X,'%ACCESSES VS TIME_OF_DAY'/
     +       1X,'-------------------------'//
     +       1X,'BROKEN DOWN AS FOLLOWS:'/
     +       1X,'1 - DISK1 IO'/
     +       1X,'I - DISK2 IO'/
     +       1X,'P - DISK2 PF'/
     +       1X,'2 - DISK2 IP'/
     +       1X,'D - TOTAL IO'//)
      WRITE(7,10)(K,K=0,9)
      WRITE(7,20)((K,K=0,9),J=0,9)
      WRITE(8,10)(K,K=0,9)
      WRITE(8,20)((K,K=0,9),J=0,9)
10    FORMAT(1X,10(9X,I1))
20    FORMAT(10X,100I1)
C
C     GET OPTIONS FROM COMMAND LINE
C
      CALL GETOPT(PERIOD,TIMES)
C
C     TOP OF MAIN LOOP
C     ----------------
C
      DO 100 I=0,TIMES
C
      CALL TIMDAT(CLOCK,6)
C
C     GET DISK DATA USING RING ZERO METER ENTRY
C
      CALL GMETR$ (G$ALL,LOC(G$BUF),BUFSIZE,CODE)
C     CALL ERRPR$ (K$SRTN,CODE,'G$ALL ',6,'GMETR$',6)
C
      CURTIM=(CLOCK(4)*060000)+(CLOCK(5)*001000)+(CLOCK(6)*001000/330)
      DELTIM=CURTIM-OLDTIM
      OLDTIM=CURTIM
      IF(DELTIM.LT.0)DELTIM=DELTIM+86400000
      DIFTIM=DELTIM/1000.0
C
C     GET USER IO VALUES
C     ==================
C
      DO 320 USER=1,CFGUSR
C
C     GET METERING DATA FOR THIS USER
C     IE TRANSFER IT FROM BIG BUFFER TO USER BUFFER
C
      INXUSR=260+(USER-1)*G$MAX4
      DO 230 K=1,G$MAX4
230   G$BUF4(K)=G$BUF(INXUSR+K)
C
C     PROCESS SYSTEM IO COUNT - CHECK FOR POSSIBLE ERRORS DUE TO LOGOUTS
C
      IF(XLOGNM.NE.'SYSTEM'.AND.BITMAP(USER).NE.1)GO TO 160 /* NOT A SYSTEM NO
C
      IF(XLOGNM.EQ.'SYSTEM')GO TO 140
C
C     A SYSTEM PHANTOM HAS LOGGED OUT
C
C     IF WE'RE LUCKY IT WON'T YET HAVE BEEN REPLACED
      IF(XLOGNM.EQ.'       '.AND.US$IOT.GE.OIOTIM(USER))GO TO 120
C
C     WE'RE UNLUCKY
      LOGFLT=.TRUE.
      GO TO 130
C
C     WE'RE LUCKY
120   SYSIO=SYSIO+US$IOT
C
130   BITMAP(USER)=0          /* THIS IS NO LONGER A SYSTEM PHANTOM
      GO TO 160
C
C     SYSTEM IS LOGGED IN
C
C     HOPEFULLY A SYSTEM PHANTOM WON'T HAVE LOGGED OUT AND IN AGAIN
C       UNDER THE SAME NUMMBER DURING THE LAST SAMPLING INTERVAL
140   IF(BITMAP(USER).EQ.0.OR.US$IOT.GE.OIOTIM(USER))GO TO 150
C
C     WE'RE UNLUCKY
      OSYSIO=OSYSIO-OIOTIM(USER)
      LOGFLT=.TRUE.
C
150   SYSIO=SYSIO+US$IOT
      BITMAP(USER)=1          /* THIS IS A SYSTEM PHANTOM
C
```

```
160   QIDTIM(USER)=US$IOT
C
C     END OF THIS USER
C     ================
C
320   CONTINUE
C
C     CALCULATE IO TIME AND ACCESS VALUES FROM METERS
C
      IF (.NOT.TELL)GO TO 340
      TIMMTR(1,1)=(DEVIOT(1,2)-ODEVIO(1,2))/330.
      TIMMTR(1,2)=(DEVIOT(1,1)-ODEVIO(1,1))/330.
      TIMMTR(1,5)=TIMMTR(1,1)+TIMMTR(1,2)
      TIMMTR(1,4)=(SYSIO-OSYSIO)/330.
      TIMMTR(1,3)=TIMMTR(1,5)-TIMMTR(1,4)
C
      DO 325 J=1,5
      TIMMTR(2,J)=TIMMTR(1,J)/DIFTIM*100
325   IF(.NOT.LOGFLT)TIMAV(J)=TIMAV(J)+TIMMTR(2,J)
C
      ACCMTR(1,1)=IOCONT(1,2)-OCNTIO(1,2)
      ACCMTR(1,4)=IOCONT(1,1)-OCNTIO(1,1)
      ACCMTR(1,5)=ACCMTR(1,1)+ACCMTR(1,4)
      ACCMTR(1,3)=NPFCN-ONPFCN
      ACCMTR(1,2)=ACCMTR(1,4)-ACCMTR(1,3)
C
      IF(ACCMTR(1,5).EQ.0)GO TO 335
      DO 330 J=1,4
      ACCMTR(2,J)=ACCMTR(1,J)/ACCMTR(1,5)*100
330   ACCAV(J)=ACCAV(J)+ACCMTR(2,J)
      ACCMTR(2,5)=100.
      ACCAV(5)=ACCAV(5)+(ACCMTR(2,1)+ACCMTR(2,2))
C
335   HOURS=CLOCK(4)/60
      MINUTS=MOD(CLOCK(4),60)
      SECNDS=CLOCK(5)
C
C     GRAPHS
C
      DISK1=TIMMTR(2,1)+0.5
      DISK2=TIMMTR(2,2)+0.5
      USR  =TIMMTR(2,3)+0.5
      SYS  =TIMMTR(2,4)+0.5
      TOTAL=TIMMTR(2,5)+0.5
      WRITE(7,700)HOURS,MINUTS,SECNDS
      WRITE(7,710)(BLANK,J=0,DISK1),ONE
      WRITE(7,710)(BLANK,J=0,DISK2),TWO
      IF(LOGFLT)GO TO 740
      WRITE(7,710)(BLANK,J=0,USR  ),U
      WRITE(7,710)(BLANK,J=0,SYS  ),S
740   IF(TOTAL.LE.100)GO TO 720
      WRITE(7,730)TOTAL
730   FORMAT('+',111X,I3)
      TOTAL=100
720   WRITE(7,710)(BLANK,J=0,TOTAL),TOT
      DISK1 =ACCMTR(2,1)+0.5
      DISK2I=ACCMTR(2,2)+0.5
      DISK2P=ACCMTR(2,3)+0.5
      DISK2 =ACCMTR(2,4)+0.5
      DISKID=ACCMTR(2,1)+ACCMTR(2,2)+0.5
      WRITE(8,700)HOURS,MINUTS,SECNDS
700   FORMAT(1X,B'##',':',B'##',':',B'##')
      WRITE(8,710)(BLANK,J=0,DISK1),ONE
      WRITE(8,710)(BLANK,J=0,DISK2I),II
      WRITE(8,710)(BLANK,J=0,DISK2P),P
      WRITE(8,710)(BLANK,J=0,DISK2),TWO
      WRITE(8,710)(BLANK,J=0,DISKID),D
710   FORMAT('+',8X,102A1)
C
C     TABLE
C
      IF(LOGFLT)WRITE(6,111)
111   FORMAT('+SYSTEM LOGOUT - SYSTEM, USER IO VALUES MAY NOT BE ',
     +       'CORRECT')
      WRITE(6,500)CLOCK(2),CLOCK(1),CLOCK(3),HOURS,MINUTS,SECNDS
500   FORMAT(/1X,'DISK USAGE',5X,A2,'-',A2,'-',A2,5X,B'##',':',B'##',
     +       ':',B'##')
      WRITE(6,510)DIFTIM
510   FORMAT(1X,'----------'//1X,'ELAPSED TIME =',F7.2,' SECS'//)
      WRITE(6,575)
      WRITE(6,520)
520   FORMAT(18X,'DISK1-IO',5X,'DISK2-IP',6X,'USER-IP',4X,'SYSTEM-IP',
     +       5X,'TOTAL-IP')
      WRITE(6,580)
      WRITE(6,585)
      WRITE(6,530)(TIMMTR(1,J),J=1,5)
530   FORMAT(2X,' SECONDS',3X,5F13.2)
      WRITE(6,590)
      WRITE(6,585)
      WRITE(6,535)(TIMMTR(2,J),J=1,5)
535   FORMAT(2X,' PERCENTAGE',5F13.1)
      WRITE(6,580)
      WRITE(6,575)
C
      WRITE(6,570)
570   FORMAT(/)
      WRITE(6,575)
      WRITE(6,540)
```

```
540   FORMAT(18X,'DISK1-IO',5X,'DISK2-IO',5X,'DISK2-PF',5X,
     +       'DISK2-IP',5X,'TOTAL-IP')
      WRITE(6,590)
      WRITE(6,595)
      WRITE(6,550)(ACCMTR(1,J),J=1,5)
550   FORMAT(3X,'ACCESSES',2X,5F13.0)
      WRITE(6,590)
      WRITE(6,595)
      WRITE(6,555)(ACCMTR(2,J),J=1,5)
555   FORMAT(3X,'PERCENTAGE',5F13.1)
      WRITE(6,590)
      WRITE(6,575)
C
575   FORMAT(2X,77('-'))
580   FORMAT('+',' ',12X,'!',26X,'!',25X,'!',11X,'!')
585   FORMAT(' ',' ',12X,'!',26X,'!',25X,'!',11X,'!')
590   FORMAT('+',' ',12X,'!',13X,'!',38X,'!',11X,'!')
595   FORMAT(' ',' ',12X,'!',13X,'!',38X,'!',11X,'!')
C
      WRITE(6,560)
560   FORMAT(12(/))
340   TELL=.TRUE.
C
      IF(I.EQ.TIMES)GO TO 100          /* FINISHED SO SKIP THE PREP
C
C     PREPARE FOR ANOTHER ROUND
C
      OSYSIO=SYSIO
      SYSIO=0
      LOGFLT=.FALSE.
      ODEVIO(1,2)=DEVIOT(1,2)
      ODEVIO(1,1)=DEVIOT(1,1)
      OCNTIO(1,2)=IOCONT(1,2)
      OCNTIO(1,1)=IOCONT(1,1)
      ONPFCN=NPFCN
C
C     DETERMINE SLEEP PERIOD
C
      CALL TIMDAT(CLOCK,6)
      DELTIM=(CLOCK(4)*060000)+(CLOCK(5)*001000)+(CLOCK(6)*001000/330)
      DELTIM=DELTIM-CURTIM
      IF(DELTIM.LT.0)DELTIM=DELTIM+86400000
      DELTIM=PERIOD-DELTIM
      CALL SLEEP$(DELTIM)
100   CONTINUE
C
C     END OF MAIN LOOP
C     ----------------
C
      WRITE(7,20)((K,K=0,9),J=0,9)
      WRITE(7,10)(K,K=0,9)
      WRITE(8,20)((K,K=0,9),J=0,9)
      WRITE(8,10)(K,K=0,9)
C
      DO 800 J=1,5
      TIMAV(J)=TIMAV(J)/TIMES
800   ACCAV(J)=ACCAV(J)/TIMES
C
C     AVERAGES
C
      WRITE(7,810)(TIMAV(J),J=1,5)
810   FORMAT(/1X,'MEAN(DISK1-IO ) =',F5.1/
     +        1X,'MEAN(DISK2-IP ) =',F5.1/
     +        1X,'MEAN(USER-IP  ) =',F5.1/
     +        1X,'MEAN(SYSTEM-IP) =',F5.1/
     +        1X,'MEAN(TOTAL-IP ) =',F5.1)
      WRITE(8,820)(ACCAV(J),J=1,5)
820   FORMAT(/1X,'MEAN(DISK1-IO) =',F5.1/
     +        1X,'MEAN(DISK2-IO) =',F5.1/
     +        1X,'MEAN(DISK2-PF) =',F5.1/
     +        1X,'MEAN(DISK2-IP) =',F5.1/
     +        1X,'MEAN(TOTAL-IO) =',F5.1)
      CALL EXIT
      END
C
      SUBROUTINE GETOPT (FREQ,TIMES)
C
      INTEGER*4 FREQ
      INTEGER*2 TIMES
      INTEGER*2 K$READ,K$IRTN,E$BPAR
      PARAMETER K$READ=1,K$IRTN=2,E$BPAR=6
      INTEGER*4 OPTION(2)
      INTEGER*2 INFO(8),CODE
C
C     Get options from command line
C
100   CALL RDTK$$ (K$READ,INFO,OPTION,4,CODE)
      IF (CODE.EQ.0.AND.INFO(1).EQ.6) RETURN /* No more options
      IF (CODE.NE.0.OR.INFO(1).NE.1) GOTO 240 /* Invalid
C
      IF (OPTION(1).EQ.'-FRE') GOTO 120 /* Frequency of report
      IF (OPTION(1).EQ.'-TIM') GOTO 140 /* Cycles/times
      GOTO 240                          /* Unknown option
C
120   CALL RDTK$$ (K$READ,INFO,OPTION,4,CODE)
      IF (CODE.NE.0.OR.INFO(1).NE.1) GOTO 240 /* Invalid
      IF (INFO(4).LT.0) GOTO 240        /* Invalid
      FREQ=INFO(4)*001000               /* Convert to milliseconds
```

```
100   CALL RDTK$$ (K$READ,INFO,OPTION,4,CODE)
      IF (CODE.EQ.0.AND.INFO(1).EQ.6) RETURN /* No more options
      IF (CODE.NE.0.OR.INFO(1).NE.1) GOTO 240 /* Invalid
C
      IF (OPTION(1).EQ.'-FRE') GOTO 120 /* Frequency of report
      IF (OPTION(1).EQ.'-TIM') GOTO 140 /* Cycles/times
      GOTO 240                          /* Unknown option
C
120   CALL RDTK$$ (K$READ,INFO,OPTION,4,CODE)
      IF (CODE.NE.0.OR.INFO(1).NE.1) GOTO 240 /* Invalid
      IF (INFO(4).LT.0) GOTO 240        /* Invalid
      FREQ=INFO(4)*001000               /* Convert to milliseconds
      GOTO 100                          /* Next option
C
140   CALL RDTK$$ (K$READ,INFO,OPTION,4,CODE)
      IF (CODE.NE.0.OR.INFO(1).NE.1) GOTO 240 /* Invalid
      IF (INFO(4).LT.0) GOTO 240        /* Invalid
      TIMES=INFO(4)
      GOTO 100                          /* Next option
C
240   CALL ERRPR$ (K$IRTN,E$BPAR,OPTION,INFO(2),'DISK ',5)
C
      WRITE (1,280)
      CALL EXIT
      RETURN                            /* If 'S' typed in
C
280   FORMAT (/,
     + 'Options available are:-',/,
     + ' ',/,
     + '   -FREQ  <secs> -- period between cycles (default = 0)',/,
     + ' ',/,
     + '   -TIMES <numb> -- number of cycles (default = 0)')
C
      END
```